

FORM PTO-1390 (Modified)
(REV 11-98)

U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE

ATTORNEY'S DOCKET NUMBER

TRANSMITTAL LETTER TO THE UNITED STATES
DESIGNATED/ELECTED OFFICE (DO/EO/US)
CONCERNING A FILING UNDER 35 U.S.C. 371

206244US2PCT

U.S. APPLICATION NO. (IF KNOWN, SEE 37 CFR

09/830588

INTERNATIONAL APPLICATION NO.

PCT/JP00/05097

INTERNATIONAL FILING DATE

01 August 2000

PRIORITY DATE CLAIMED

09 September 1999

TITLE OF INVENTION

ACCESS METHOD AND RECORDING MEDIUM HAVING ACCESS PROCESSING PROGRAM

APPLICANT(S) FOR DO/EO/US

Takashi MATSUMOTO

Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:

1. ☒ This is a **FIRST** submission of items concerning a filing under 35 U.S.C. 371.
2. ☐ This is a **SECOND** or **SUBSEQUENT** submission of items concerning a filing under 35 U.S.C. 371.
3. ☒ This is an express request to begin national examination procedures (35 U.S.C. 371(f)) at any time rather than delay examination until the expiration of the applicable time limit set in 35 U.S.C. 371(b) and PCT Articles 22 and 39(1).
4. ☐ A proper Demand for International Preliminary Examination was made by the 19th month from the earliest claimed priority date.
5. ☒ A copy of the International Application as filed (35 U.S.C. 371 (c) (2))
 - a. ☐ is transmitted herewith (required only if not transmitted by the International Bureau).
 - b. ☒ has been transmitted by the International Bureau.
 - c. ☐ is not required, as the application was filed in the United States Receiving Office (RO/US).
6. ☒ A translation of the International Application into English (35 U.S.C. 371(c)(2)).
7. ☒ A copy of the International Search Report (PCT/ISA/210).
8. ☒ Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. 371 (c)(3))
 - a. ☐ are transmitted herewith (required only if not transmitted by the International Bureau).
 - b. ☐ have been transmitted by the International Bureau.
 - c. ☐ have not been made; however, the time limit for making such amendments has NOT expired.
 - d. ☒ have not been made and will not be made.
9. ☐ A translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)).
10. ☒ An oath or declaration of the inventor(s) (35 U.S.C. 371 (c)(4)).
11. ☐ A copy of the International Preliminary Examination Report (PCT/IPEA/409).
12. ☐ A translation of the annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371 (c)(5)).

Items 13 to 20 below concern document(s) or information included:

13. ☐ An Information Disclosure Statement under 37 CFR 1.97 and 1.98.
14. ☐ An assignment document for recording. A separate cover sheet in compliance with 37 CFR 3.28 and 3.31 is included.
15. ☒ A **FIRST** preliminary amendment.
16. ☐ A **SECOND** or **SUBSEQUENT** preliminary amendment.
17. ☐ A substitute specification.
18. ☐ A change of power of attorney and/or address letter.
19. ☐ Certificate of Mailing by Express Mail
20. ☒ Other items or information:

Request for Consideration of Documents Cited in International Search Report

Notice of Priority

PCT/IB/304

Drawings (8 Sheets)

U.S. APPLICATION NO. (IF KNOWN, SEE 37 CFR 1.53) 09/830588	INTERNATIONAL APPLICATION NO. PCT/JP00/05097	ATTORNEY'S DOCKET NUMBER 206244US2PCT
--	--	---

21. The following fees are submitted:				CALCULATIONS PTO USE ONLY	
BASIC NATIONAL FEE (37 CFR 1.492 (a) (1) - (5)) :					
<input type="checkbox"/> Neither international preliminary examination fee (37 CFR 1.482) nor international search fee (37 CFR 1.445(a)(2)) paid to USPTO and International Search Report not prepared by the EPO or JPO \$1,000.00					
<input checked="" type="checkbox"/> International preliminary examination fee (37 CFR 1.482) not paid to USPTO but International Search Report prepared by the EPO or JPO \$860.00					
<input type="checkbox"/> International preliminary examination fee (37 CFR 1.482) not paid to USPTO but international search fee (37 CFR 1.445(a)(2)) paid to USPTO \$710.00					
<input type="checkbox"/> International preliminary examination fee paid to USPTO (37 CFR 1.482) but all claims did not satisfy provisions of PCT Article 33(1)-(4) \$690.00					
<input type="checkbox"/> International preliminary examination fee paid to USPTO (37 CFR 1.482) and all claims satisfied provisions of PCT Article 33(1)-(4) \$100.00					
ENTER APPROPRIATE BASIC FEE AMOUNT =				\$860.00	
Surcharge of \$130.00 for furnishing the oath or declaration later than <input type="checkbox"/> 20 <input type="checkbox"/> 30 months from the earliest claimed priority date (37 CFR 1.492 (e)).				\$0.00	
CLAIMS	NUMBER FILED	NUMBER EXTRA	RATE		
Total claims	10 - 20 =	0	x \$18.00	\$0.00	
Independent claims	4 - 3 =	1	x \$80.00	\$80.00	
Multiple Dependent Claims (check if applicable). <input type="checkbox"/>				\$0.00	
TOTAL OF ABOVE CALCULATIONS =				\$940.00	
Reduction of 1/2 for filing by small entity, if applicable. Verified Small Entity Statement must also be filed (Note 37 CFR 1.9, 1.27, 1.28) (check if applicable). <input type="checkbox"/>				\$0.00	
SUBTOTAL =				\$940.00	
Processing fee of \$130.00 for furnishing the English translation later than <input type="checkbox"/> 20 <input type="checkbox"/> 30 months from the earliest claimed priority date (37 CFR 1.492 (f)).				\$0.00	
TOTAL NATIONAL FEE =				\$940.00	
Fee for recording the enclosed assignment (37 CFR 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 CFR 3.28, 3.31) (check if applicable). <input type="checkbox"/>				\$0.00	
TOTAL FEES ENCLOSED =				\$940.00	
				Amount to be refunded	\$
				charged	\$

- ☒ A check in the amount of **\$940.00** to cover the above fees is enclosed.
- ☐ Please charge my Deposit Account No. _____ in the amount of _____ to cover the above fees.
A duplicate copy of this sheet is enclosed.
- ☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. **15-0030** A duplicate copy of this sheet is enclosed.

NOTE: Where an appropriate time limit under 37 CFR 1.494 or 1.495 has not been met, a petition to revive (37 CFR 1.137(a) or (b)) must be filed and granted to restore the application to pending status.

SEND ALL CORRESPONDENCE TO:



22850

Surinder Sachar
Registration No. 34,423

SIGNATURE

Marvin J. Spivak

NAME

24,913

REGISTRATION NUMBER

DATE

May 9 2001

DOCKET NO. 206244US2PCT

IN RE APPLICATION OF: Takashi MATSUMOTO

SERIAL NO.: New U.S. PCT Application (Based on PCT/JP00/05097)

09 / 8 3 0 5 8 8

FILED: HEREWITH

FOR: ACCESS METHOD AND RECORDING MEDIUM HAVING ACCESS PROCESSING PROGRAM

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

Sir:

Transmitted herewith is an amendment in the above-identified application.

- ☒ No additional fee is required.
- ☐ Small entity status of this application under 37 C.F.R. §1.9 and §1.27 has been established by a verified statement previously submitted.
- ☐ Small entity status of this application under 37 C.F.R. §1.9 and §1.27 has been established by a verified statement submitted herewith.
- ☒ Additional documents filed herewith: PCT Transmittal Letter/Notice of Priority/Drawings (8 Sheets)
English Translation of Specification/Request for Consideration/PCT/IB/304/Declaration/Preliminary Amendment
International Search Report/Check for \$940.00

The fee has been calculated as shown below.

(Col. 1)		(Col. 2)		(Col. 3)	SMALL ENTITY		OTHER THAN A SMALL ENTITY	
	CLAIMS REMAINING AFTER		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDITIONAL FEE	RATE	ADDITIONAL FEE
TOTAL	* 10	MINUS	** 20	= 0	X9 =	\$	X18 =	\$.00
INDEP	* 4	MINUS	*** 4	= 0	X40 =	\$	X80 =	\$.00
<input type="checkbox"/> FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+135=	\$	+270=	\$
TOTAL						\$	TOTAL	\$.00

A check in the amount of \$_____ is attached.

- XX Please charge any additional fees for the papers being filed herewith and for which no check is enclosed herewith, or credit any overpayment to deposit Account No. 15-0030. A duplicate copy of this sheet is enclosed.
- XX If these papers are not considered timely filed by the Patent and Trademark Office, then a petition is hereby made under 37 C.F.R. §1.136, and any additional fees required under 37 C.F.R. §1.136 for any necessary extension of time may be charged to deposit Account No. 15-0030. A duplicate copy of this sheet is enclosed.



22850

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak

Marvin J. Spivak
Attorney of Record
Registration No. 24,913
Surinder Sachar
Registration No. 34,423

(703) 413-3000

*If the entry in Column 2 is less than the entry in Column 1 write "0" in Column 3.
**If the "Highest Number Previously paid for" IN THIS SPACE is less than 20 write "20" in this space.
***If the "Highest Number Previously paid for" IN THIS SPACE is less than 3 write "3" in this space.7/93

2062444US

09/830588

JC08 Rec'd PCT/PTO 09 MAY 2007

IN THE UNITED STATES PATENT & TRADEMARK OFFICE

IN RE APPLICATION OF :
TAKASHI MATSUMOTO : ATTN: APPLICATION DIVISION
SERIAL NO: NEW U.S. PCT APPLN :
(Based on PCT/JP00/05097)
FILED: HEREWITH :
FOR: ACCESS METHOD AND :
RECORDING MEDIUM HAVING
ACCESS PROCESSING PROGRAM

PRELIMINARY AMENDMENT

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

SIR:

Prior to a first examination on the merits, please amend the above-identified application as follows:

IN THE SPECIFICATION

Please enter the attached Substitute Specification. A marked-up copy showing the changes made in the Substitute Specification is also provided.

IN THE CLAIMS

Please amend the claims as follows:

1. (Amended) An access method comprising the steps of:

requesting, by a process, an open procedure of an I/O device or an I/O interface to an operating system;

allocating, by the operating system, a context identifier for indicating a request storing area of the process, and further mapping a memory page corresponding to the context identifier as an accessing address to a pending register indicating that there is an unprocessed request;

writing, by the process, contents of operation requests to the I/O device or the I/O interface into the request storing area;

notifying, by the process, the I/O device or the I/O interface that there is an unprocessed request by use of the accessing address for the pending register; and

reading, by the I/O device or the I/O interface, the operation requests of the process which are stored at the location related to the context identifier stored in the pending register.

3. (Amended) The access method according to claim 1, comprising the steps, performed by the I/O device or the I/O interface, of:

identifying that the pending register has been accessed;

obtaining the physical address of the request storing area of the process having accessed the pending register by referring to the embedded memory storing the physical address of the request storing area of each process based on the context identifier; and

reading out the contents of the request storing area, and of realizing the operation requests.

4. (Amended) The access method according to claim 1,

wherein the physical address specifying the I/O device or the I/O interface includes a function select field indicating a position of the pending register and a context identifier field indicating the process, and

an address decoder stores the context identifier of the physical address in the pending register in the case where a fixed address indicating the position of the pending register is stored in the function select field of the physical address.

5. (Amended) The access method according to claim 1, wherein other data corresponding to the context identifier is stored in the pending register according to needs.

6. (Amended) The access method according to claim 1, wherein, in any one of the cases where the process requests a close procedure of the I/O device or the I/O interface to the operating system and where the process is ended, the operating system withdraws the physical page for the pending register allocated to the process, withdraws the context identifier for the I/O device or the I/O interface, and/or clears an entry of the physical address to the request area of the process on the embedded memory.

7. (Amended) A recording medium having an access-processing program,

wherein the access-processing program includes the steps of:

requesting, by a process, an open procedure of an I/O device or an I/O interface to an operating system;

allocating, by the operating system, a context identifier for indicating a request storing area of the process, and further mapping a memory page corresponding to the context identifier as an accessing address to a pending register indicating that there is an unprocessed request;

writing, by the process, contents of operation requests to the I/O device or the I/O interface into the request storing area;

notifying, by the process, the I/O device or the I/O interface that there is an unprocessed request by use of the accessing address for the pending register; and

reading, by the I/O device or the I/O interface, the operation requests of the process which are stored at the location related to the context identifier stored in the pending register.

Please add new claims 8-10 as follows:

8. (New) An address decoding method for selecting a physically same object device in a computer system, the method comprising the steps of:

selecting an object device using only a part of a physical address for decoding;

using another part of the physical address as an identifier for distinguishing multiple physical addresses selecting the object device; and

using the identifier in order to identify an application process having accessed the object device.

9. (New) A computer system having an address decoder selecting a physically same object device, comprising:

means for selecting an object device by use of only a part of a physical address for decoding the address;

means for identifying an identifier by use of another part of the physical address as the identifier for distinguishing multiple physical addressed selecting the object device; and

means for identifying an application process having accessed the object device by use of the identifier.

10. (New) The computer system according to claim 9, further comprising:

means for executing requests corresponding to the identified application process by the object device based on contents of access.

REMARKS

Favorable consideration of this application, as presently amended, is respectfully requested.

The present Preliminary Amendment is submitted to place the above-identified application in more proper format under United States practice.


By the present Preliminary Amendment the specification has been amended to correct for minor informalities and to clarify the disclosure. The amendments to the specification are presented in the attached Substitute Specification. The changes made to the specification are also shown in the attached Comparison version showing the changes made to the original specification which are presented in the Substitute Specification. The changes made to the specification are deemed to be self-evident from the original disclosure, and thus are not deemed to raise any issues of new matter.

The claims have been amended by the present response to clarify the recitations therein and to add new claims 8-10 for examination. The claim amendments and the submission of new claims 8-10 are not deemed to raise any issues of new matter.

The present application is believed to be in condition for a full and thorough examination on the merits. An early and favorable consideration of the present application is hereby respectfully requested.

Respectfully submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.



Gregory J. Maier
Attorney of Record
Registration No. 25,599
Surinder Sachar
Registration No. 34,423



22850

(703) 413-3000
Fax No.: (703) 413-2220
GJM:SNS\la
I:\atty\SNS\206244-pr.wpd

Marked-Up Copy
Serial No: 206244US
Amendment Filed on:

IN THE SPECIFICATION

Please enter the attached substitute specification.

IN THE CLAIMS

Please amend the claims as follows:

--1. (Amended) An access method comprising the steps of:

requesting, by a [first] process, an open procedure of an I/O device or an I/O interface to an operating system;

allocating, by the operating system, a context identifier for indicating a request storing area of the [first] process, and further mapping a memory page corresponding to the context identifier as an accessing address to a pending register indicating that there is an unprocessed request;

writing, by the [first] process, contents of operation requests to the I/O device or the I/O interface into the request storing area;

notifying, by the [operating system] process, the I/O device or the I/O interface that there is an unprocessed request by use of the accessing address for the pending register; and

reading, by [any one of] the I/O device or the I/O interface, the [request of the first process based on] operation requests of the process which are stored at the location related to the context identifier stored in the pending register.

3. (Amended) The access method according to [any one of claims 1 and 2] claim 1, comprising the steps, performed by the I/O device or the I/O interface, of:

identifying that the pending register has been accessed;

obtaining the physical address of the request storing area of the process having accessed the pending register by referring to the embedded memory storing the physical address of the request storing area of each process based on the context identifier; and

reading out the contents of the request storing area, and of realizing the [request contents] operation requests.

4. (Amended) The access method according to [any one of claims 1 to 3] claim 1,

wherein the physical address specifying the I/O device or the I/O interface includes a function select field indicating a position of the pending register and a context identifier field indicating the process, and

an address decoder stores the context identifier of the physical address in the pending register in the case where a fixed address indicating the position of the pending register is stored in the function select field of the physical address.

5. (Amended) The access method according to [any one of claims 1 to 4] claim 1, wherein other data corresponding to the context identifier is stored in the pending register according to needs.

6. (Amended) The access method according to [any one of claims 1 to 5] claim 1, wherein, in any one of the cases where the [first] process requests a close procedure of the I/O device or the I/O interface to the operating system and where the [first] process is ended, the operating system withdraws the [address] physical page for the pending register allocated to the [first] process, withdraws the context identifier for the I/O device or the I/O interface,

and/or clears an entry of the physical address to the request area of the [first] process on the embedded memory.

7. (Amended) A recording medium having an access-processing program,

wherein the access-processing program includes the steps of:

requesting, by a [first] process, an open procedure of an I/O device or an I/O interface to an operating system;

allocating, by the operating system, a context identifier for indicating a request storing area of the [first] process, and further mapping a memory page corresponding to the context identifier as an accessing address to a pending register indicating that there is an unprocessed request;

writing, by the [first] process, contents of operation requests to the I/O device or the I/O interface into the request storing area;

notifying, by the [operating system] process, the I/O device or the I/O interface that there is an unprocessed request by use of the accessing address for the pending register; and

reading, by the I/O device or the I/O interface, the [request of the first process based on] operation requests of the process which are stored at the location related to the context identifier stored in the pending register.--

Claims 8-10 (Added).

09/830588

JC08 Rec'd PCT/PTO 09 MAY 2001

ACCESS METHOD
AND
RECORDING MEDIUM HAVING ACCESS PROCESSING
PROGRAM

5

Field of Invention

The present invention relates to an access method and a recording medium having an access-processing program therein, particularly to a user-level I/O access method and an access-processing program in a virtualized multi-context environment.

10

Description of the Related Art

Fig. 6 shows an explanatory view of a conventional system 1. This figure shows an explanatory view of a system where control registers of I/O devices are mapped into the conventional kernel address space.

15

As shown in the figure, in the conventional system 1, control registers of an I/O device are set to be accessible only by a kernel mode under a general-purpose operating system (OS), thus preventing an illegal operation by a user application. A physical address space is divided into a kernel address space and a user address space. The I/O device can be accessed only by entry addresses in the kernel space, which correspond to the I/O device. And, when a process A as a user application wants to use the I/O device, the process A issues an operation request through a system call of the OS. Then, the OS checks validity of the operation request in the kernel mode, and thereafter, executes a concrete operation to the control registers in the I/O device.

20

25

30

Moreover, recently, a system has been gradually used, in which the control registers of the I/O device are mapped into an

address space of a user application, and the user application (user process) directly accesses the control registers and reduces an overhead of the I/O operations. Fig. 7 shows an explanatory view of a conventional system 2. This figure shows a constitution example of the system where control registers are mapped into the user address space. In this conventional system 2, the process A maps a physical page for the I/O device into the user address space of the process A. From the page for I/O device, the control registers of the I/O device can be used.

Next, Fig. 8 shows an explanatory view of a conventional system 3. This figure shows an example of a system where the operation request is transferred from the user address space by direct memory access (DMA). In order to eliminate or reduce the overhead of the system call, it is necessary that control information concerning the I/O device is stored into the pages in the user address space. In this point of view, the conventional system 3 is the same as the conventional system 2. Note that, if the control registers of the I/O device are merely mapped into the user address space, there occurs confusion when the control registers are accessed from the plurality of user applications. For solving this, contents of requests for the I/O device can be stored on a memory, in order that the I/O device reads the contents of requests and set them on the control registers. Specifically, the I/O device updates the contents of the control registers by DMA. And if the I/O device operates so as not to start processing a next content of request before the processing for the preceding request is finished (or reached to a breakpoint), the confusion of the operations does not occur. Moreover, since a memory resource is protected under the general-purpose OS, there is no risk of overwriting the contents of requests by other applications.

What becomes important in the conventional system 3 is how to indicate the location of the operation requests on the memory to

the I/O device. In the conventional system 3, a context register storing a pointer indicating the location of the requests on the memory is provided in the I/O device. The OS grasps which user's process is running. And, the OS can rewrite the context register according to the running process. In the case where a certain application (for example, process A) is allocated to the processor, a storage location for the operation requests from the concerned application is set in the context register.

In the conventional system 3, a pending register which notifies the registration of the operation requests is mapped into the user address space according to each application. The pending register is used as a flag indicating that a process requires operations by the I/O device. In pages for user processes, the operation requests are stored. The contents of the requests are, for example, an I/O command. If the I/O device is a printer, the I/O command is a print command, a page-feed command or the like. If the I/O device is the other I/O device than the printer, the I/O command is a command as to which device input/output are performed to. When the pending register is accessed, the I/O device fetches (using DMA) the operation requests from the memory location indicated by the context register (the pointer), and executes the requests.

Summary of the Invention

However, in the conventional system 1, there has been a problem that the overhead of I/O operation is very large since a series of the kernel programs including the system call must be called out for each time when the user application tries to perform an I/O operation.

Therefore, the conventional system 2 has been proposed. However, in this conventional system 2, multiple applications

cannot use the I/O device simultaneously. In the example of Fig. 7, when the process A is using the I/O device, the process B cannot use the I/O device. Moreover, since one application changes the contents of the control register of the I/O device solely at its discretion, consistency in the operation of the control register collapses if multiple applications operate the control register on time-sharing basis. Accordingly, in the conventional system 2, an I/O resource must be allocated to one application exclusively for use.

Therefore, the conventional system 3 has been proposed as a system for solving the problem of the overhead of the system call in the conventional system 1 and the problem of the multiple allocation of resources in the conventional system 2. However, in the conventional system 3, the context register in the I/O device becomes a part of a processor context, and for each time when the OS performs a context switch, a location pointer for the operation requests of an application allocated to the processor must be updated. Since the context switch is one of the fundamental operations of the OS, it is difficult to execute modification of extending the processor context to the existing OS. Still less, the processor context cannot be extended by adding a device driver. Moreover, in the case where a computer system connecting with an I/O device is of a multi-processor configuration, the context registers are required in the number equal to that of the processors, and any access to the pending register needs to be discernible as to which processor it is from. In the case where a control circuit of the I/O device is integrated into an LSI, in consideration of multi-processor correspondence, many context registers must be prepared in advance. This results in a waste of cost in a typical single processor or a parallel system with a small number of processors.

As described above, the user-level access methods to I/O devices, which have been heretofore implemented, have the

problems to be solved, respectively. Moreover, for example, there has been performed the method of mapping control registers for I/O operations into a user memory space in order to speed up the access to I/O devices at the user level as described above. However, according to the conventional systems, the multiple applications cannot use the devices simultaneously.

In consideration of the foregoing points, the present invention provides an access method enabling multiple applications to execute simultaneously I/O (or I/O interface) operations at a low cost by exploiting a memory management unit, and provides a recording medium for recording an access-processing program.

One of the features of the present invention is that a memory alias (a state where multiple physical addresses are allocated to a single object) is generated intentionally, and that an accessed application is identified by a mechanism detecting by which alias address it has been accessed. Each alias of the same object is aligned to a page-boundary. Since physical memory of the host system is managed by a page unit, allocating a different alias for each application inhibits an illegal action such as an access to another alias address.

According to solutions of the present invention, provided is an access method and a recording medium having an access-processing program comprising the steps of:

requesting, by a user process, an open procedure of an I/O device (or I/O interface) to the operating system;

allocating, by the operating system, a context identifier for indicating a request storing area of the process, and further mapping a memory page corresponding to the context identifier as an accessing address to a pending register indicating that there is an unprocessed request;

writing, by the process, contents of operation requests to the I/O device into the request storing area;

notifying, by the process, the I/O device or the I/O interface that there is an unprocessed request by use of the accessing address for the pending register; and

reading, by the I/O device, the operation requests of the process which are stored at the location related to the context identifier in the pending register.

Furthermore, there is provided a recording medium for recording an access-processing program.

In addition, there is provided an address decoding method for selecting a physically same object device in a computer system, the method comprising the steps of:

selecting an object device using only a part of a physical address for decoding;

using another part of the physical address as an identifier for distinguishing multiple physical addresses selecting the object device; and

using the identifier in order to identify an application process having accessed the object device.

Furthermore, there is provided a computer system having an address decoder selecting a physically same object device, comprising:

means for selecting an object device by use of only a part of a physical address for decoding the address;

means for identifying an identifier by use of another part of the physical address as the identifier for distinguishing multiple physical addresses selecting the object device; and

means for identifying an application process having accessed the object device by use of the identifier.

Brief Description of the Drawings

Fig. 1 is a constitutional view of a system relating to the

present invention.

Fig. 2 is an explanatory view of an access method according to the present invention.

Fig. 3 is a flowchart regarding an OS and a process in the access method according to the present invention.

Fig. 4 is a flowchart regarding an I/O device in the access method according to the present invention.

Fig. 5 is an explanatory view of a mechanism storing data in a pending register.

Fig. 6 is an explanatory view of a conventional system 1.

Fig. 7 is an explanatory view of a conventional system 2.

Fig. 8 is an explanatory view of a conventional system 3.

Description of the Preferred Embodiments

The present invention provides a novel access method to an I/O device at a user level, which solves the foregoing problems.

In the conventional system 3, if the I/O device can identify which application has executed an access to the pending register indicating that the operation requests not subjected to the I/O processing are present on the memory, a low-cost method of obtaining the memory location for the operation requests can be constituted. In an extreme instance, if sufficient pairs of the context registers and the pending registers exist, and if each application space maps one pending register, then an address for reading out the requests can be decided by use of the contents of the context registers corresponding to the accessed pending registers. In this case, the context registers need not to be provided in the I/O device, and areas on the embedded memory in the I/O device can be sufficiently substituted therefor. Positions of the context registers on the memory can be computed from the positions of the accessed pending registers. If the context register

can be provided as an area on the embedded memory inside the I/O device, and if the number of pending registers can be restricted into one or a small number, a user-level access method to an I/O device with very low cost and very little overhead or non-overhead can be achieved.

In the present invention, only one substance of the pending register is prepared. However, multiple address aliases of the pending register are prepared, and a mechanism is prepared for determining from which alias the access is made to the pending register, thus the application (processor context) that has been accessing the pending register is identified. Hereinbelow, description will be made for an embodiment of the present invention by showing a concrete example.

Fig. 1 shows an example of a constitutional view of the system relating to the present invention.

Fig. 1 shows a system for realizing an access method to the I/O device with a low overhead at the user level by utilizing, for example, a workstation or personal computer with a standard configuration.

This system comprises: I/O devices (A) 1-1 and (B) 1-2; a processor 2; a cache 3; a system LSI 4; a main memory 5; and an I/O bus 6. The I/O devices (A) 1-1 and (B) 1-2 comprise the pending registers indicating that unprocessed requests are registered therein. These pending registers are mapped into a memory space (or subjected to memory mapping), and are accessed from the processor (to be described later in detail). The processor 2 executes various processing according to the OS in cooperation with the cache 3. The cache 3 is a high-speed memory for copying a program and data for the purpose of speeding up the processing. The system LSI 4 controls transmission/receipt of the data interactively among the I/O devices (A) 1-1 and (B) 1-2, the processor 2 and the main memory 5. The main memory 5 stores

the program, the data and the like, such as the operating system (OS), various processes as applications and the physical address space. And the I/O bus 6 can be configured as an extension bus.

Next, Fig. 2 shows an explanatory view of the access method according to the present invention. This figure shows an address map of pending registers.

In this example, processes A to C 210 to 212, a physical address space 22 and an I/O device 1 are shown. The I/O device 1 comprises: a pending register 23; a DMA engine 24; a request executing register 25; and an embedded memory 26. Moreover, pages for process A to C 270 to 272 and memory pages for process A to C 280 to 282 are prepared in the physical address space, and the requests from the respective processes are stored in request storing areas 290 to 292.

Fig. 3 shows a flowchart regarding the OS and the process in the access method according to the present invention. And, Fig. 4 shows a flowchart regarding the I/O device in the access method according to the present invention. Hereinbelow, description will be made for the access method of the present invention with reference to Figs. 2 to 4.

First, the processing regarding the OS and the process will be described based on Fig. 2 and the flowchart of Fig. 3.

Since the OS manages the process, the OS has grasped the presence of the processes A to C from the beginning when they were created. What is understood from an open procedure is that the process that requested the open procedure to the OS is going to use the I/O device 1 later. First, the process for using the I/O device, from the viewpoint of process, will be executed as follows.

In Step S101, a process (for example, process A 210) requests the open procedure of the I/O device 1 to the OS. At this time, the process instructs the OS on an area where its own requests to the I/O device 1 will be stored. Next, in Step S102, first during the

open procedure, the OS allocates an unused context ID for the I/O device 1 (for example, ID = 0) to the process A 210, and maps the physical memory page 280 (page 0) corresponding to this ID as an logical address for access to the pending register 23 by the process A 210. The mapped logical address is notified from the OS to the process A 210 as a result of the open procedure (system call). Moreover, in Step S102, secondly during the open procedure, the OS stores a pointer (physical address) to the request storing area 290 of the process A 210 in the embedded memory 26 in the I/O device 1. The stored address for requests is set in a location easily computable with the context ID allocated to the process A 210. That is, a table for retrieving the pointer to the request storing area out of the context ID is formed on the embedded memory.

In Step S103, the process A 210 describes the contents of operation requests to the I/O device 1 into its own request storing area 290. In Step S104, after description of the contents of requests, the process A 210 notifies the I/O device of presence of the unprocessed requests by use of the logical address for the pending register 23, which is allocated by the open procedure. To be concrete, the process A 210 accesses the address of the pending register 23. In Step S105, the I/O device reads the request out of the request storing area 290. In Step S106, if the process A 210 has another request to the I/O device 1, Step 103 and Step 104 are iterated. In Step S107, when the process A requests a close procedure of the I/O device to the OS, or when the process A 210 is ended, the OS withdraws the physical page for the pending register 23 allocated to the process A 210, withdraws the context ID for the I/O device (to be stored as a blank ID), and clears the entry of the pointer to the storing area 290 of the request from process A 210 on the embedded memory 26.

Next, based on Fig. 2 and the flowchart of Fig. 4, the processing regarding the I/O device will be described. From the

viewpoint of the I/O device, the processing will be executed as follows.

In Step S201, it is identified from output of a first-in first-out (FIFO) memory output that the pending register 23 has been
 5 accessed. In Step S202, since the FIFO output includes the context ID, the physical address of the request storing area 290 of the process A 210 that has accessed the pending register is identified by referring to the table on the embedded memory 26. In Step S203, the DMA engine 24 reads out the contents of the
 10 request storing area 290, controls the internal register 25 according to the contents, and executes the request. In Step S204, the I/O device iterates these operations until, for example, the power source is cut off.

The process B and the process C also can use the I/O device 1
 15 in exactly the same manner. Here, it is important from a viewpoint of reduction of the overhead that the OS is never interposed in Steps S103 and S104. Even if the processes allocated to the host processor are switched on time-sharing basis, the requesting process is uniquely understood by use of the physical
 20 address to the pending register that includes the context ID. Accordingly, the OS needs not to perform any additional operation in the event of switching the processes. Note that, in the conventional system 3, the context registers must have been switched.

25 Next, Fig. 5 shows an explanatory view of a mechanism storing data in the pending register.

A memory structure on the top of Fig. 5 shows a physical address 51 which specifies the I/O device.

Here, as an example, the physical address 51 is constituted
 30 of 36 bits and includes: a field for function select as page offset in 0 to 11th bits; a field for the context ID for process in 12th to 19th bits; and a field for function select in 20th to 35th bits. Moreover, as in

the figure, a field for indicating the context ID (e.g., 8 bits in the figure) is prepared in a portion higher in order than the page offset (e.g., 12 bits or 13 bits in a typical processor). It should be noted that the field for the context ID is not used for decoding the address, and that what is usable for decoding the address of the pending register is portions of the field higher in order than the context ID and the page offset. The area for the function select indicates a location of the pending register.

The function select in the high-order address must be decoded (decided uniquely), while the low-order address (offset portion) does not have to be decoded. In the case where the offset portion is not to be decoded, the pending register is specified only by the high-order address, regardless of what kind of pattern the offset portion is. Note that, since the memory address can be allocated to the process only by page unit, other processes are not affected if the decoding of the offset portion is omitted. On the contrary, when the decoding is not omitted, another register or a memory can be allocated on the same page of the pending register. In this case, an address of a different bit pattern in the page offset portion is used.

Moreover, the context ID field indicates the process using the I/O device (for example, any one of the processes A to C). When the context ID field has 8 bits, even if the 256 processes issue requests for using the I/O device (to open) simultaneously, the context ID field can cope with all requests.

In the case where a fixed address indicating the location of the pending register is stored in the function select field of the physical address 51, an address decoder 52 for the pending register selects the pending register, and stores the context ID of the physical address 51 in the pending register 53. Moreover, according to needs, data 54 can be stored in the pending register 53 in correspondence with the context ID. The data 54 can be given

from, for example, the processor, the main memory, an interface and the like (for example, in the event of store/swap access). An output 55 of the pending register 53 includes the context ID, and occasionally, also includes the data corresponding thereto.

5 Each application has a specified context ID allocated thereto, and maps only the page according to the context ID. Specifically, where the application is different, the same pending register is accessed by a physical address having a different context ID field. Where the I/O device has a capability of holding and interpreting
10 the context ID field each time when it is accessed, the I/O device can clarify as to which application has accessed the pending register. The access to the pending register is an access by the user application, and there is no limitation to the timing of execution. The I/O processing cannot be performed without
15 synchronizing the operation of the I/O device and the receipt of a new request by the pending register. Therefore, the access to the pending register 53 must be buffered to the FIFO configuring the same. At this time, the contents of access (information telling that they are "writing" and "written data" when the contents are
20 writing) are not only buffered, but the context ID is also buffered simultaneously. The address of the area, where the operation requests are stored by the application, and the address are obtained from the context ID by an appropriate calculation, and the I/O device reads out the requests using DMA engine 24, and
25 performs the input/output processing.

A return-value or interruption can be used to deal with overfilling of the FIFO buffer.

• In the case where the return-value is used to deal with the overfilling, if the "written data" is not required, the pending
30 register is accessed by a load (memory reading-out) instruction. And in the case where the FIFO buffer is overfilled, an error code (for example, -1) is returned as the return-value. The user

application checks the return-value and iterates the access until it succeeds. If the "written data" is required, the user application delivers the data to the I/O device 1 while receiving the return-value by a swap (atomic read-write operation) instruction.

5 • It takes a long time and a much cost for the processor to execute the load instruction or the swap instruction through the extension bus for the I/O device in comparison with the store instruction. Therefore, in the case where the interruption is used to deal with the overfilling of the FIFO buffer, the overfilling of the FIFO buffer is detected by the interruption in order to enable the access to the pending register by the store instruction. If the capacity of the FIFO buffer is sufficient and the capability of the I/O device 1 is not saturated, then the overfilling of the FIFO buffer does not occur. If the overfilling of the FIFO buffer occurs, then, thinking that it costs inevitably a certain amount of the processing cost, the interruption is generated to temporarily store the application which have failed to access and the contents of access in the pending register by the OS. Thereafter, when the application and the contents of access are registerable in the I/O device, they are processed.

Note that, the access method according to the present invention can be applied to an appropriate access to an interface, not being limited to the I/O device. Moreover, the access method according to the present invention can be provided as an access-processing program by a recording medium such as CD-ROM or a transmission medium such as the Internet.

Industrial Applicability

30 As described above, according to the present invention, there can be provided the access method enabling multiple applications to execute simultaneously the I/O operations (or the interface

2

ACCESS METHOD
AND
RECORDING MEDIUM HAVING ACCESS PROCESSING
PROGRAM

5

Field of Invention

The present invention relates to an access method and a recording medium having an access-processing program therein, particularly to a user-level I/O access method and an access-processing program in a virtualized multi-context environment.

Description of the Related Art

Fig. 6 shows an explanatory view of a conventional system 1. This figure shows an explanatory view of a system mapped into the conventional kernel address space.

As shown in the figure, in the conventional system 1, control registers of an I/O device are set to be accessible only by a kernel mode under a general-purpose operating system (OS), thus preventing an illegal operation by a user application. A physical address space is divided into a kernel address space and a user address space. The I/O device can be used only in the case where an entry in the kernel address space, which corresponds to the I/O device, is accessed. And, when a process A as a user application wants to use the I/O device, the process A issues an operation request through a system call of the OS. Then, the OS checks validity of the operation request in the kernel mode, and thereafter, executes a concrete operation to the control registers in the I/O device.

Moreover, recently, a system has been gradually used, in which the control registers of the I/O device are mapped into an

address space of a user application, and the user application (user process) directly accesses the control registers and reduces an overhead of the I/O operations. Fig. 7 shows an explanatory view of a conventional system 2. This figure shows a constitution example of the system mapped into the user address space. In this conventional system 2, the process A maps a page for the I/O device, which is present in the physical address space, into the user address space. From the page for I/O device, the control registers of the I/O device can be used.

Next, Fig. 8 shows an explanatory view of a conventional system 3. This figure shows an example of a system where the operation request is obtained from the user address space by direct memory access (DMA). In order to eliminate or reduce the overhead of the system call in the conventional system 1, it is necessary that control information concerning the I/O device is mapped into the user address space. In this point of view, the conventional system 3 is the same as the conventional system 2. Note that, if the control registers of the I/O device are merely mapped into the user address space, there occurs confusion when the control registers are accessed from the plurality of user applications. For solving this, contents of requests for the I/O device can be stored on a memory, in order that the I/O device reads the contents of requests and set them on the control registers. Specifically, the I/O device updates the contents of the control registers by DMA. And if the I/O device operates so as not to start processing a next content of request before the processing for a certain content of request is finished (or reached to a breakpoint), the confusion of the operations does not occur. Moreover, since a memory resource is protected under the general-purpose OS, there is no risk of overwriting the contents of requests by other applications.

What becomes important here is how to indicate the location

of the contents of requests on the memory to the I/O device. In the conventional system 3, a context register storing a pointer indicating the location of the contents of requests on the memory is provided in the I/O device. The context register is a register
 5 indicating, for example, which process task is running in a host processor. The OS grasps which user's task (process) is running. And, the OS can rewrite the context register according to the running process. In the case where a certain application (for example, process A) is allocated to the processor, a storage location
 10 for the request contents from the concerned application is set in the context register.

Moreover, a pending register which notifies the registration of the contents of requests is mapped into the user address space according to each application. The pending register is, for
 15 example, a flag indicating that a certain process accesses the I/O device. In a page for the process A, the requests are stored. The contents of the requests are, for example, an I/O command. If the I/O device is a printer, the I/O command is a print command, a page-feed command or the like. If the I/O device is the other I/O
 20 device than the printer, the I/O command is a command as to which device input/output are performed to. Moreover, in the conventional system 3, when the pending register is accessed, the I/O device fetches (using DMA) the concerned contents of requests from the memory location indicated by the context register (the
 25 pointer), and executes the requests.

Summary of the Invention

However, in the conventional system 1, there has been a
 30 problem that the overhead is large since a series of the kernel programs including the system call must be called out for each time when the user application tries to perform an I/O operation.

Therefore, the conventional system 2 has been proposed. However, in this conventional system 2, multiple applications cannot use the I/O device simultaneously. In this example, when the process A is using the I/O device, the process B cannot use the I/O device. Moreover, since one application changes the contents of the control register of the I/O device solely at its discretion, consistency in the operation of the control register collapses if multiple applications operate the control register on time-sharing basis. Accordingly, in the conventional system 2, an I/O resource must be allocated to one application exclusively for use.

Therefore, the conventional system 3 has been proposed as a system for solving the problem of the overhead of the system call in the conventional system 1 and the problem of the allocation in the conventional system 2. However, in the conventional system 3, the context register in the I/O device becomes a part of a processor context, and for each time when the OS performs a context switch, a request content storage location for the application allocated to the processor must be updated. Since the context switch is one of the fundamental operations of the OS, it is difficult to execute modification of extending the processor context to the existing OS. Still less, the processor context cannot be extended by adding a device driver. Moreover, in the case where a computer system connecting with an I/O device is of a multi-processor configuration, the context registers are required in the number equal to that of the processors, and any access to the pending register needs to be discernible as to which processor it is from. In the case where a control circuit of the I/O device is integrated into an LSI, in consideration of multi-processor correspondence, many context registers must be prepared in advance. This results in a waste of cost in a typical single processor or a parallel system with a small number of processors.

As described above, the user-level access methods to I/O

devices, which have been heretofore implemented, have the problems to be solved, respectively. Moreover, for example, there has been performed the method of mapping a register for communication and I/O operations into a user memory space in order to speed up the access to communication and I/O devices at the user level as described above. However, according to the conventional systems, the multiple applications cannot use the devices simultaneously.

In consideration of the foregoing points, the present invention provides an access method enabling multiple applications to execute simultaneously communication and I/O operations and interface operations at a low cost by exploiting a memory management unit, and provides a recording medium for recording an access-processing program.

One of the features of the present invention is that a memory alias by a page unit (a state where multiple addresses are allocated to physically the same object) is generated intentionally, and that an accessed application is identified by a mechanism detecting by which alias address it has been accessed. Since the memory is managed by a page unit, allocating a different alias for each application inhibits an illegal action such as an access to another alias address.

According to solutions of the present invention, provided is requesting, by a first process, an open procedure of an I/O device or an I/O interface to an operating system;

allocating, by the operating system, a context identifier for indicating a request storing area of the first process, and further mapping a memory page corresponding to the context identifier as an accessing address to a pending register indicating that there is an unprocessed request;

writing, by the first process, contents of requests to the I/O device or the I/O interface into the request storing area;

notifying, by the operating system, the I/O device or the I/O interface that there is an unprocessed request by use of the accessing address for the pending register; and

reading, by any one of the I/O device or the I/O interface, the request of the first process based on the context identifier stored in the pending register. Furthermore, there is provided a recording medium for recording an access-processing program.

Brief Description of the Drawings

Fig. 1 is a constitutional view of a system relating to the present invention.

Fig. 2 is an explanatory view of an access method according to the present invention.

Fig. 3 is a flowchart regarding an OS and a process in the access method according to the present invention.

Fig. 4 is a flowchart regarding an I/O device in the access method according to the present invention.

Fig. 5 is an explanatory view of a mechanism storing data in a pending register.

Fig. 6 is an explanatory view of a conventional system 1.

Fig. 7 is an explanatory view of a conventional system 2.

Fig. 8 is an explanatory view of a conventional system 3.

Description of the Preferred Embodiments

The present invention provides a novel access method to an I/O device at a user level, which solves the foregoing problems.

In the conventional system 3, if the I/O device side can identify which application has executed an access to the pending register indicating that the requests not subjected to the I/O processing are present on the memory, a method of obtaining a

memory address of the contents of requests can be constituted. In an extreme instance, if sufficient pairs of the context registers and the pending registers exist, and if each application space maps one pending register, then an address for reading out the requests can be decided by use of the contents of the context registers corresponding to the accessed pending registers. In this case, the context registers need not to be provided in the I/O device, and areas on the embedded memory in the I/O device can be sufficiently substituted therefor. Positions of the context registers on the memory can be computed from the positions of the accessed pending registers. If the context register can be provided as an area on the embedded memory inside the I/O device, and if the number of pending registers can be restricted into one or a small number, a user-level access method to an I/O device with very low cost and very little overhead or non-overhead can be achieved.

In the present invention, only one substance of the pending register is prepared. However, multiple address aliases of the pending register are prepared, and a mechanism is prepared for determining from which alias the access is made to the pending register, thus the application (processor context) that has been accessing the pending register is identified. Hereinbelow, description will be made for an embodiment of the present invention by showing a concrete example.

Fig. 1 shows an example of a constitutional view of the system relating to the present invention.

Fig. 1 shows a system for realizing an access method to the I/O device with a low overhead at the user level by utilizing, for example, a workstation or personal computer with a standard configuration.

This system comprises: I/O devices (A) 1-1 and (B) 1-2; a processor 2; a cache 3; a system LSI 4; a main memory 5; and an I/O bus 6. The I/O devices (A) 1-1 and (B) 1-2 comprise the pending

registers indicating that unprocessed requests are registered therein. These pending registers are mapped into a memory space (or subjected to memory mapping), and are accessed from the processor (to be described later in detail). The processor 2
 5 executes various processing according to the OS in cooperation with the cache 3. The cache 3 is a high-speed memory for copying a program and data for the purpose of speeding up the processing. The system LSI 4 controls transmission/receipt of the data interactively among the I/O devices (A) 1-1 and (B) 1-2, the
 10 processor 2 and the main memory 5. The main memory 5 stores the program, the data and the like, such as the operating system (OS), various processes as applications and the physical address space. And the I/O bus 6 can be configured as an extension bus.

Next, Fig. 2 shows an explanatory view of the access method according to the present invention. This figure shows an address
 15 map of pending registers.

In this example, processes A to C 210 to 212, a physical address space 22 and an I/O device 1 are shown. The I/O device 1 comprises: a pending register 23; a DMA engine 24; a request
 20 executing register 25; and an embedded memory 26. Moreover, pages for process A to C 270 to 272 and memory pages for process A to C 280 to 282 are prepared in the physical address space, and the requests from the respective processes are stored in request storing areas 290 to 292.

Fig. 3 shows a flowchart regarding the OS and the process in the access method according to the present invention. And, Fig. 4 shows a flowchart regarding the I/O device in the access method according to the present invention. Hereinbelow, description will
 25 be made for the access method of the present invention with reference to Figs. 2 to 4.

First, the processing regarding the OS and the process will be described based on Fig. 2 and the flowchart of Fig. 3.

Since the OS manages the process, the OS has grasped the presence of the processes A to C from the beginning when they were created. What is understood from an open procedure is that the process that requested the open procedure to the OS is going to use the I/O device 1 later. First, the process for using the I/O device, from the viewpoint of process, will be executed as follows.

In Step S101, a process (for example, process A 210) requests the open procedure of the I/O device 1 to the OS. At this time, the process instructs the OS on an area where its own requests to the I/O device 1 will be stored. Next, in Step S102, first during the open procedure, the OS allocates an unused context ID for the I/O device 1 (for example, ID = 0) to the process A 210, and maps the memory page 280 (page 0) corresponding to this ID as an address for access to the pending register 23 by the process A 210. The mapped logical address is notified from the OS to the process A 210 as a result of the open procedure. Moreover, in Step S102, secondly during the open procedure, the OS stores a pointer (physical address) to the request storing area 290 of the process A 210 in the embedded memory 26 in the I/O device 1. The stored address is set in a location easily computable with the context ID allocated to the process A 210. That is, a table for retrieving the pointer to the request storing area out of the context ID is formed on the embedded memory.

In Step S103, the process A 210 describes the contents of requests to the I/O device 1 into its own request storing area 290. In Step S104, after description of the contents of requests, the OS notifies the I/O device of presence of the unprocessed requests by use of the address for the pending register 23, which is allocated by the open procedure. To be concrete, the process A 210 accesses the address of the pending register 23. In Step S105, the I/O device reads the request out of the request storing area 290. In Step S106, if the process A 210 has another request to the I/O device 1, Step

103 and Step 104 are iterated. In Step S107, when the process A requests a close procedure of the I/O device to the OS, or when the process A 210 is ended, the OS withdraws the address allocated to the process A 210, withdraws the context ID for the I/O device (to be stored as a blank ID), and clears the entry of the pointer to the storing area 290 of the request from process A 210 on the embedded memory 26.

Next, based on Fig. 2 and the flowchart of Fig. 4, the processing regarding the I/O device side will be described. From the viewpoint of the I/O device, the processing will be executed as follows.

In Step S201, it is identified from output of a first-in first-out (FIFO) memory output that the pending register 23 has been accessed. In Step S202, since the FIFO output includes the context ID, the physical address of the request storing area 290 of the process A 210 that has accessed the pending register is identified by referring to the table on the embedded memory 26. In Step S203, the DMA engine 24 reads out the contents of the request storing area 290, controls the internal register 25 according to the contents, and executes the request. In Step S204, the I/O device iterates these operations until, for example, the power source is cut off.

The process B and the process C also can use the I/O device 1 in exactly the same manner. Here, it is important from a viewpoint of reduction of the overhead that the OS is never interposed in Steps S103 and S104. Even if the processes allocated to the host processor are switched on time-sharing basis, the requesting process is uniquely understood by use of the address to the pending register that includes the context ID. Accordingly, the OS needs not to perform any additional operation in the event of switching the processes. Note that, in the conventional system 3, the context registers must have been switched.

Next, Fig. 5 shows an explanatory view of a mechanism storing data in the pending register.

A memory structure on the top of Fig. 5 shows a physical address 51 which specifies the I/O device.

Here, as an example, the physical address 51 is constituted of 36 bits and includes: a field for function select as page offset in 0 to 11th bits; a field for the context ID for process in 12th to 19th bits; and a field for function select in 20th to 35th bits. Moreover, as in the figure, a field for indicating the context ID (e.g., 8 bits in the figure) is prepared in a portion higher in order than the page offset (e.g., 12 bits or 13 bits in a typical processor). It should be noted that the field for the context ID is not used for decoding the address, and that what is usable for decoding the address of the pending register is portions of the field higher in order than the context ID and the page offset. The area for the function select indicates a location of the pending register.

The function select in the high-order address must be decoded (decided uniquely), while the low-order address (offset portion) does not have to be decoded. In the case where the offset portion is not to be decoded, the pending register is specified only by the high-order address, regardless of what kind of pattern the offset portion is. Note that, since the address can be allocated to the process only by page unit, other processes are not affected if the decoding of the offset portion is omitted. On the contrary, when the decoding is not omitted, another register or a memory can be allocated on the same page of the pending register. In this case, an address of a different bit pattern in the page offset portion is used.

Moreover, the context ID field indicates the process using the I/O device (for example, any one of the processes A to C). When the context ID field has 8 bits, even if the 256 processes issue requests for using the I/O device (to open) simultaneously, the context ID field can cope with all requests.

In the case where a fixed address indicating the location of the pending register is stored in the function select field of the physical address 51, an address decoder 52 for the pending register selects the pending register, and stores the context ID of the physical address 51 in the pending register 53. Moreover, according to needs, data 54 can be stored in the pending register 53 in correspondence with a process identifier. The data 54 can be given from, for example, the processor, the main memory, an interface and the like (for example, in the event of store/swap access). An output 55 of the pending register 53 includes the context ID, and occasionally, also includes the data corresponding thereto.

Each application has a specified context ID allocated thereto, and maps only the page according to the context ID. Specifically, where the application is different, the same pending register is accessed by an address having a different context ID field. Where the I/O device has a capability of holding and interpreting the context ID field each time when it is accessed, the I/O device can clarify as to which application has accessed the pending register. The access to the pending register is an access by the user application, and there is no limitation to the timing of execution. The I/O processing cannot be performed without synchronizing the operation of the I/O device and the receipt of a new request by the pending register. Therefore, the access to the pending register 53 must be buffered to the FIFO configuring the same. At this time, the contents of access (information telling that they are "writing" and "written data" when the contents are writing) are not only buffered, but the context ID is also buffered simultaneously. The address space, where the request contents are stored by the application, and the address are obtained from the context ID by an appropriate calculation, and the I/O device reads out the request contents, and performs the input/output processing.

A return-value or interruption can be used to deal with overfilling of the FIFO buffer.

• In the case where the return-value is used to deal with the overfilling, if the "written data" is not required, the pending register is accessed by a load (memory reading-out) instruction. And in the case where the FIFO buffer is overfilled, an error code (for example, -1) is returned as the return-value. The user application checks the return-value and iterates the access until it succeeds. If the "written data" is required, the user application delivers the data to the I/O device 1 while receiving the return-value by a swap (atomic read-write operation) instruction.

• It takes a long time and a much cost for the processor to execute the load instruction or the swap instruction through the extension bus for the I/O device in comparison with the store instruction. Therefore, in the case where the interruption is used to deal with the overfilling of the FIFO buffer, the overfilling of the FIFO buffer is detected by the interruption in order to enable the access to the pending register by the store instruction. If the capacity of the FIFO buffer is sufficient and the capability of the I/O device 1 is not saturated, then the overfilling of the FIFO buffer does not occur. If the overfilling of the FIFO buffer occurs, then, thinking that it costs inevitably a certain amount of the processing cost, the interruption is generated to temporarily store the application which have failed to access and the contents of access in the pending register by the OS. Thereafter, when the application and the contents of access are registerable in the I/O device, they are processed.

Note that, the access method according to the present invention can be applied to an appropriate access to an interface, not being limited to the I/O device. Moreover, the access method according to the present invention can be provided as an access-processing program by a recording medium such as CD-ROM or a

transmission medium such as the Internet.

Industrial Applicability

5 As described above, according to the present invention, there
can be provided the access method enabling multiple applications
to execute simultaneously the communication and I/O operations
and the interface operation at a low cost by exploiting the memory
management unit, and the recording medium for recording the
10 access-processing program.

CLAIMS

1. An access method comprising the steps of:

requesting, by a first process, an open procedure of an I/O
5 device or an I/O interface to an operating system;

allocating, by the operating system, a context identifier for
indicating a request storing area of the first process, and further
mapping a memory page corresponding to the context identifier as
an accessing address to a pending register indicating that there is
10 an unprocessed request;

writing, by the first process, contents of requests to the I/O
device or the I/O interface into the request storing area;

notifying, by the operating system, the I/O device or the I/O
interface that there is an unprocessed request by use of the
15 accessing address for the pending register; and

reading, by any one of the I/O device or the I/O interface, the
request of the first process based on the context identifier stored in
the pending register.

2. The access method according to claim 1, wherein the operating
20 system stores a physical address of the request storing area
corresponding to each process into an embedded memory in the I/O
device or the I/O interface.

3. The access method according to any one of claims 1 and 2,
25 comprising the steps, performed by the I/O device or the I/O
interface, of:

identifying that the pending register has been accessed;

obtaining the physical address of the request storing area of
30 the process having accessed the pending register by referring to the
embedded memory storing the physical address of the request
storing area of each process based on the context identifier; and

reading out the contents of the request storing area, and of realizing the request contents.

4. The access method according to any one of claims 1 to 3,

5 wherein the physical address specifying the I/O device or the I/O interface includes a function select field indicating a position of the pending register and a context identifier field indicating the process, and

10 an address decoder stores the context identifier of the physical address in the pending register in the case where a fixed address indicating the position of the pending register is stored in the function select field of the physical address.

15 5. The access method according to any one of claims 1 to 4, wherein other data corresponding to the context identifier is stored in the pending register according to needs.

20 6. The access method according to any one of claims 1 to 5, wherein, in any one of the cases where the first process requests a close procedure of the I/O device or the I/O interface to the operating system and where the first process is ended, the operating system withdraws the address allocated to the first process, withdraws the context identifier for the I/O device or the I/O interface, and/or clears an entry of the physical address to the request area of the
25 first process on the embedded memory.

7. A recording medium having an access-processing program,

30 wherein the access-processing program includes the steps of: requesting, by a first process, an open procedure of an I/O device or an I/O interface to an operating system;

 allocating, by the operating system, a context identifier for indicating a request storing area of the first process, and further

mapping a memory page corresponding to the context identifier as an accessing address to a pending register indicating that there is an unprocessed request;

5 writing, by the first process, contents of requests to the I/O device or the I/O interface into the request storing area;

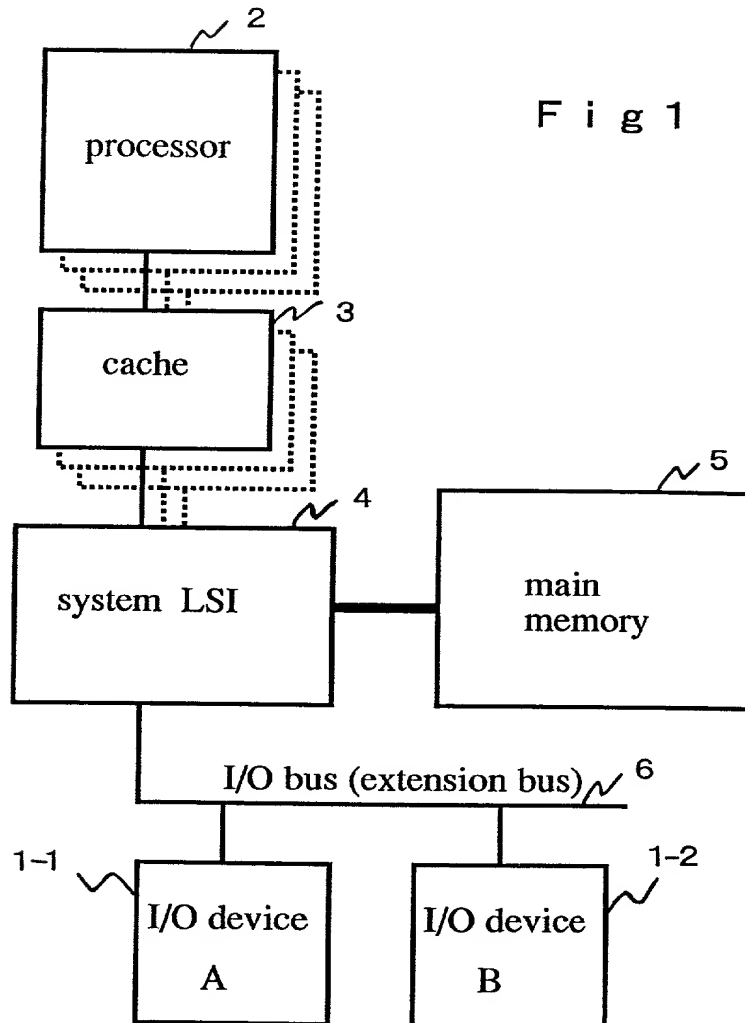
notifying, by the operating system, the I/O device or the I/O interface that there is an unprocessed request by use of the accessing address for the pending register; and

10 reading, by the I/O device or the I/O interface, the request of the first process based on the context identifier stored in the pending register.

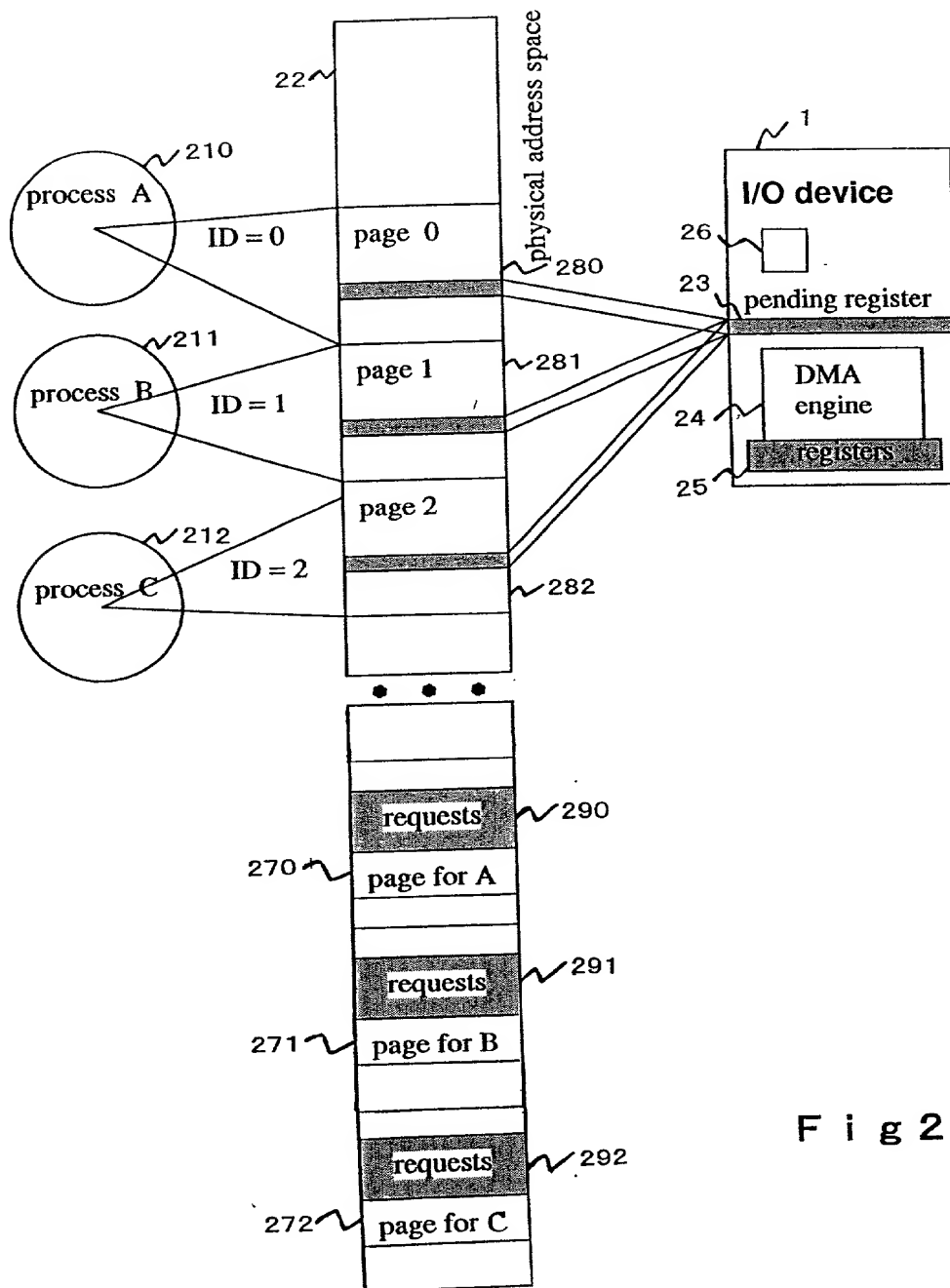
ABSTRACT

Multiple applications enable communications, I/O operation and an I/O interface operations simultaneously at a low cost. The process A requests an OS to allocate an area where a request to an I/O device is put. The OS also allocates an unused context ID for the I/O device to the process A, maps a memory page corresponding to the context ID as an address for accessing the pending register for the process A, and stores a pointer (a physical address) to a request storing area of the process A into an embedded memory in the I/O device. The process A writes contents of requests in its own request storing area, and the OS notifies the I/O device that there is an unprocessed request by use of the address for the pending register. The I/O device reads out the contents of the request storing area by a DMA engine, and realizes the request.

1/8



2/8



F i g 2

3/8

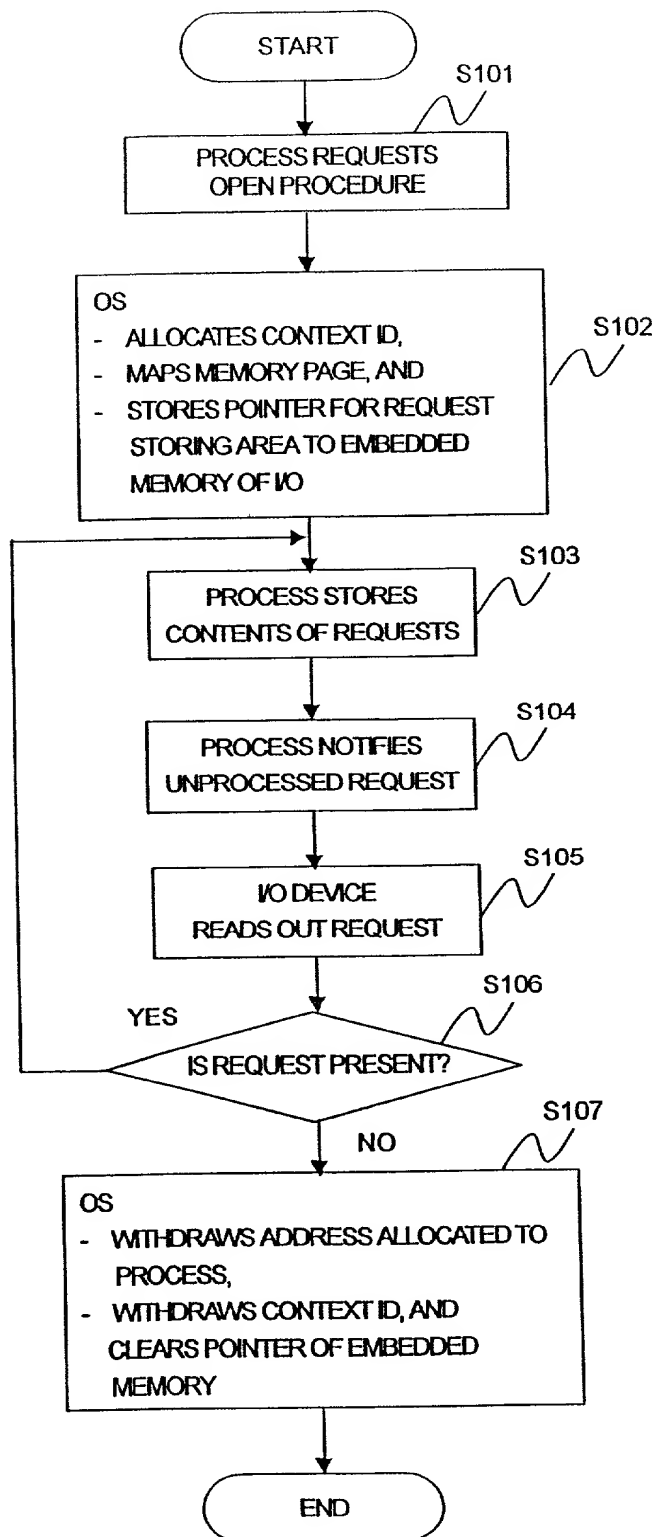
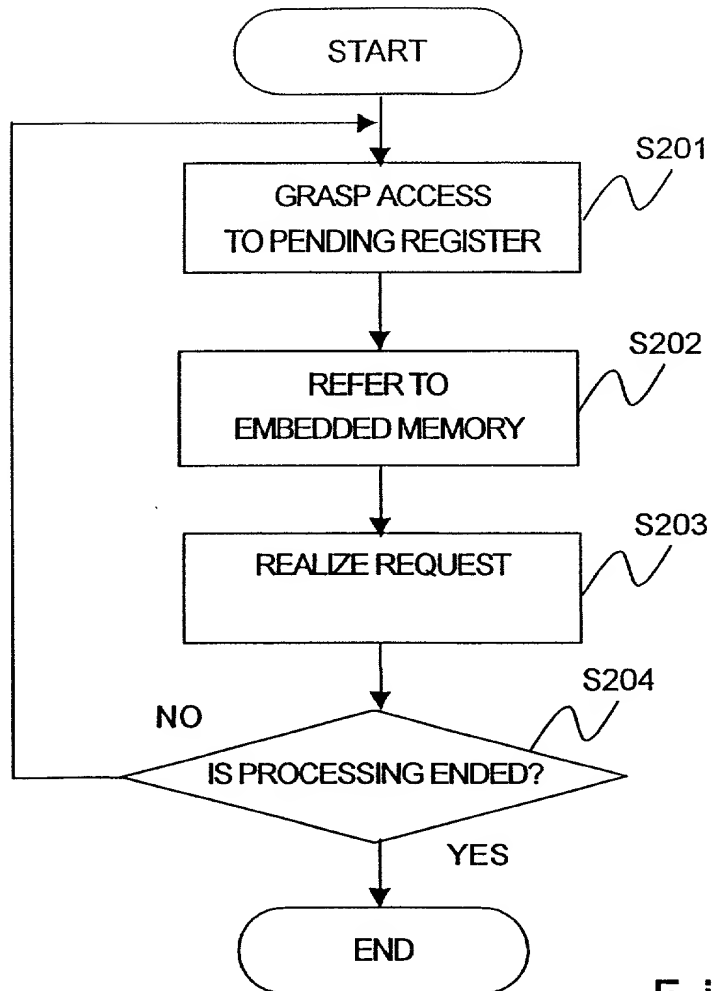


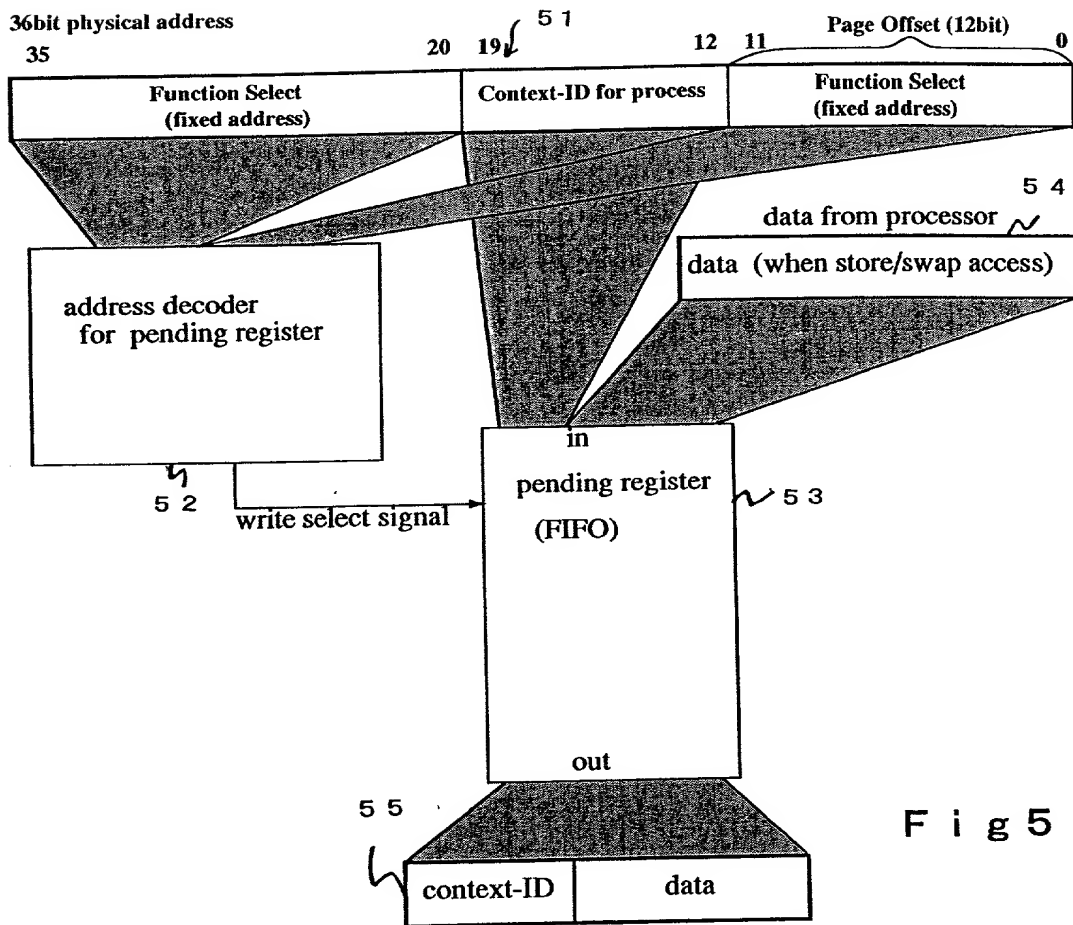
Fig 3

4/8



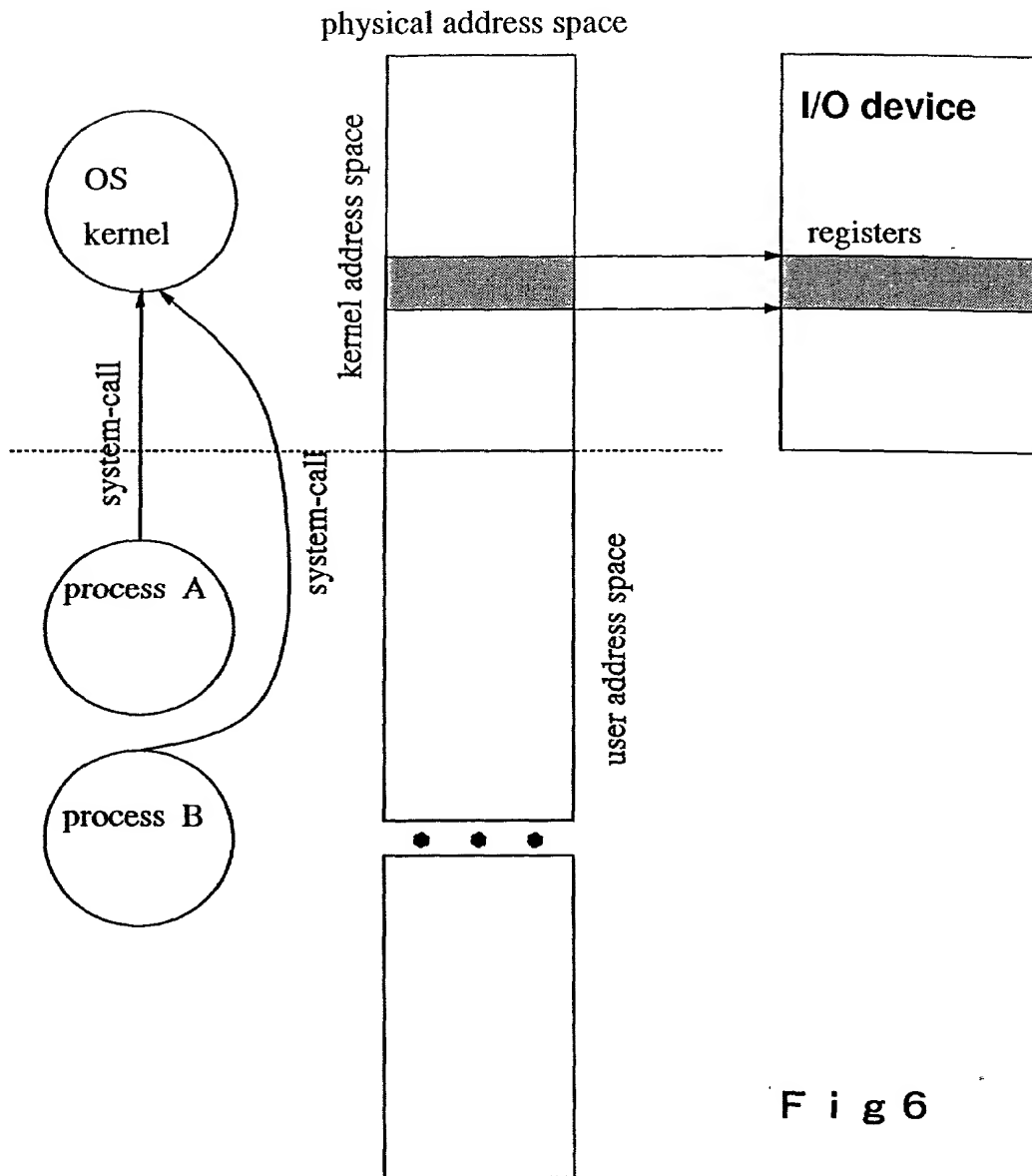
F i g 4

5/8

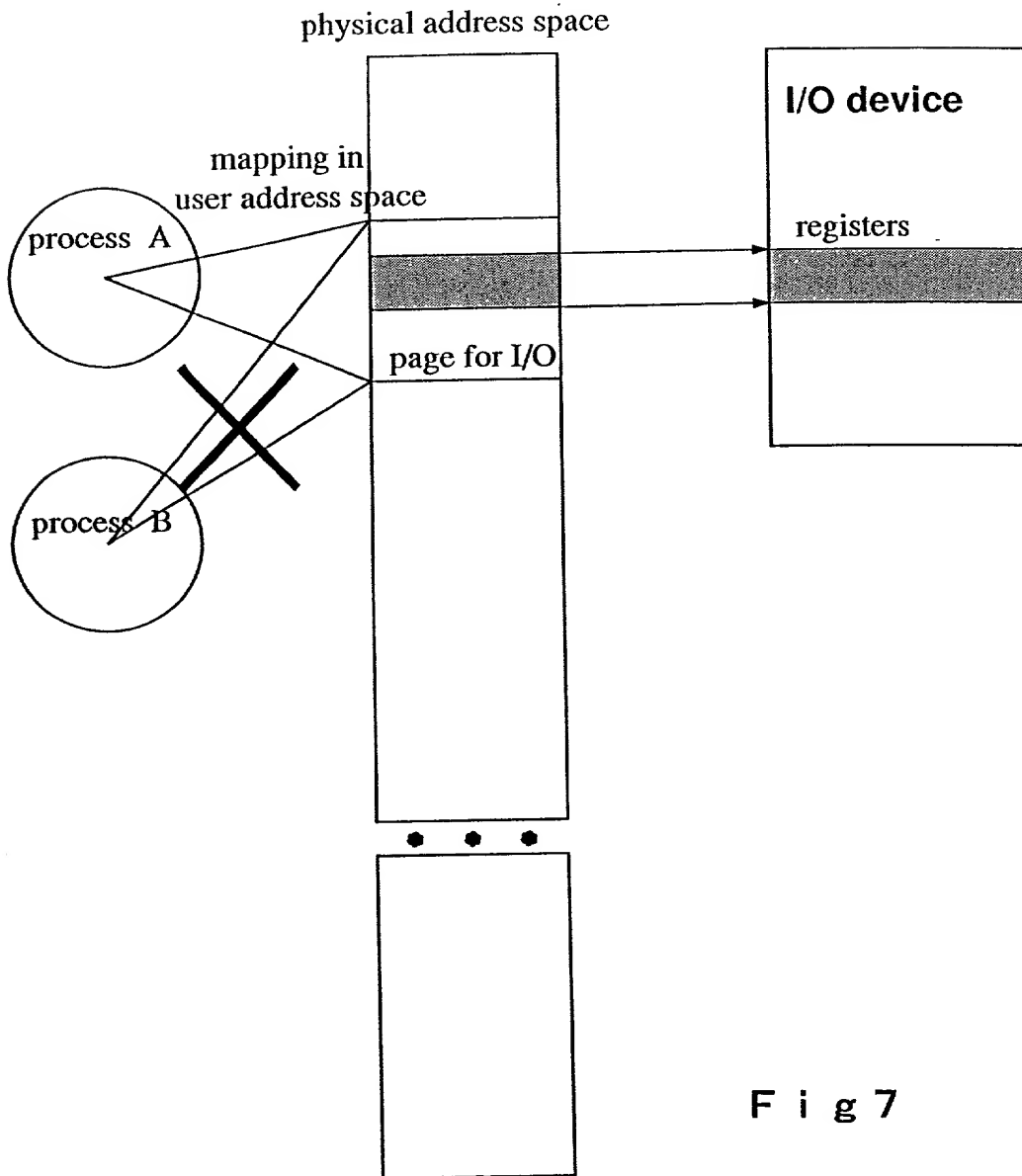


F i g 5

6/8

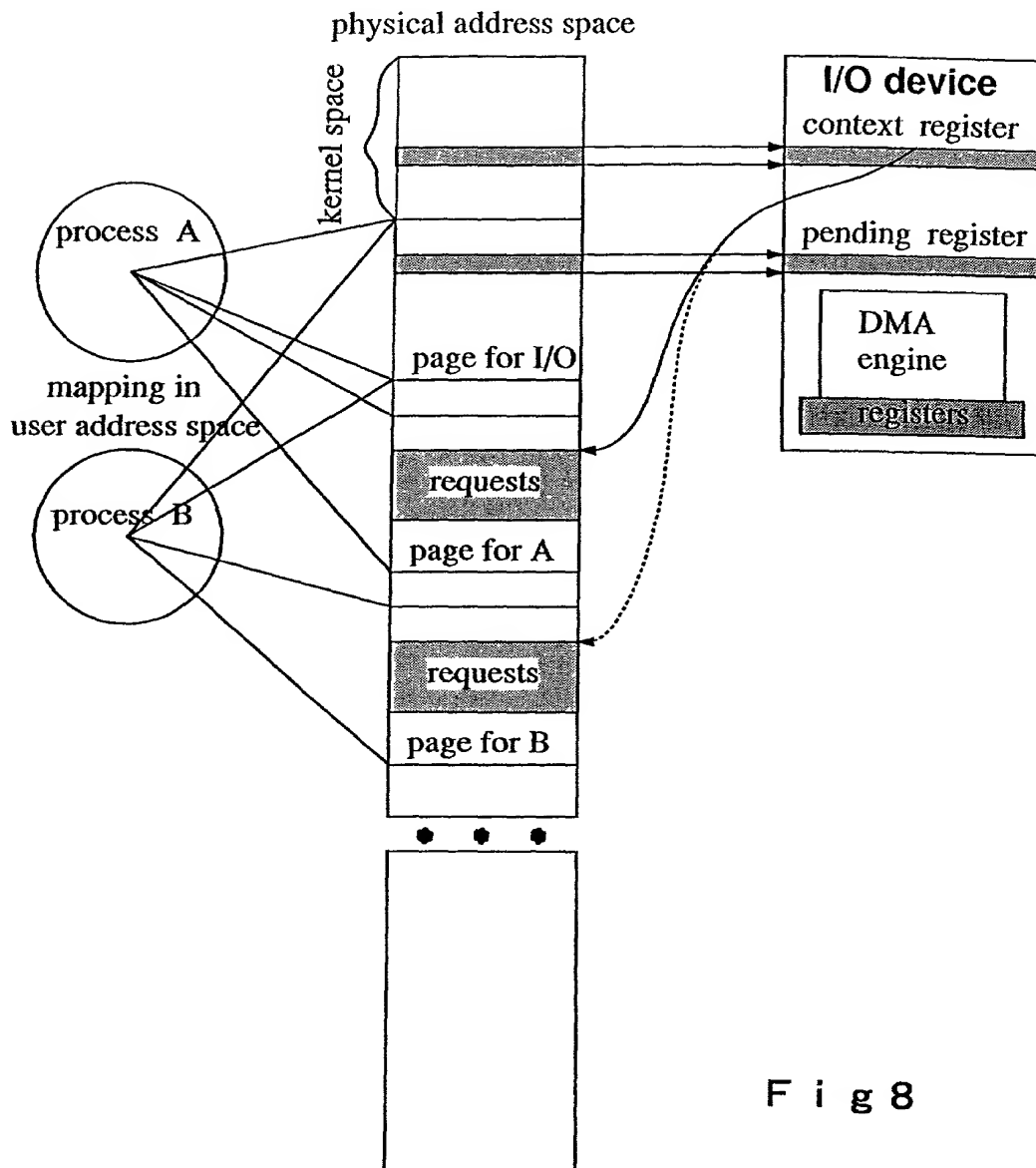


7/8



F i g 7

8/8



Declaration and Power of Attorney For Patent Application

206244US2PCT

特許出願宣言書及び委任状

Japanese Language Declaration

日本語宣言書

下記の氏名の発明者として、私は以下の通り宣言します。

As a below named inventor, I hereby declare that:

私の住所、私書箱、国籍は下記の私の氏名の後に記載された通りです。

My residence, post office address and citizenship are as stated next to my name.

下記の名称の発明に関して請求範囲に記載され、特許出願している発明内容について、私が最初かつ唯一の発明者（下記の氏名が一つの場合）もしくは最初かつ共同発明者（下記の名称が複数の場合）であると信じています。

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled.

ACCESS METHOD AND RECORDING

MEDIUM HAVING ACCESS PROCESSING

PROGRAM

the specification of which

☐ is attached hereto.

☒ was filed on 01 August 2000

as United States Application Number or

PCT International Application Number

PCT/JP00/05097 and was amended on

 (if applicable).

上記発明の明細書は、

☐ 本書に添付されています。

☐ 月 日に提出され、米国出願番号または特許協定条約国際出願番号を とし、
(該当する場合) に訂正されました。

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

私は、特許請求範囲を含む上記訂正後の明細書を検討し、内容を理解していることをここに表明します。

私は、連邦規則法典第37編第1条56項に定義されるとおり、特許資格の有無について重要な情報を開示する義務があることを認めます。

Japanese Language Declaration

(日本語宣言書)

私は、米国法典第35編119条 (a) - (d) 項又は365条 (b) 項に基づき下記の、米国以外の国の少なくとも一カ国を指定している特許協力条約365 (a) 項に基づく国際出願、又は外国での特許出願もしくは発明者証の出願についての外国優先権をここに主張するとともに、優先権を主張している、本出願の前に出願された特許または発明者証の外国出願を以下に、枠内をマークすることで、示しています。

Prior Foreign Application(s)
外国での先行出願

<u>11-255272</u>	<u>JAPAN</u>
(Number) (番号)	(Country) (国名)
<u> </u>	<u> </u>
(Number) (番号)	(Country) (国名)

私は、第35編米国法典119条 (e) 項に基づいて下記の米国特許出願規定に記載された権利をここに主張いたします。

<u> </u>	<u> </u>
(Application No.) (出願番号)	(Filing Date) (出願日)

私は、下記の米国法典第35編120条に基づいて下記の米国特許出願に記載された権利、又は米国を指定している特許協力条約365条 (c) に基づく権利をここに主張します。また、本出願の各請求範囲の内容が米国法典第35編112条第1項又は特許協力条約で規定された方法で先行する米国特許出願に開示されていない限り、その先行米国出願書提出日以降で本出願書の日本国内または特許協力条約国際提出日までの期間中に入手された、連邦規則法典第37編1条56項で定義された特許資格の有無に関する重要な情報について開示義務があることを認識しています。

<u>PCT/JP00/05097</u>	<u>01 AUGUST 2000</u>
(Application No.) (出願番号)	(Filing Date) (出願日)

<u> </u>	<u> </u>
(Application No.) (出願番号)	(Filing Date) (出願日)

私は、私自信の知識に基づいて本宣言書中で私が行なう表明が真実であり、かつ私の入手した情報と私の信じているところに基づく表明が全て真実であると信じていること、さらに故意になされた虚偽の表明及びそれと同等の行為は米国法典第18編第1001条に基づき、罰金または拘禁、もしくはその両方により処罰されること、そしてそのような故意による虚偽の声明を行なえば、出願した、又は既に許可された特許の有効性が失われることを認識し、よってここに上記のごとく宣誓を致します。

I hereby claim foreign priority under Title 35, United States Code, Section 119 (a)-(d) or 365(b) of any foreign application(s) for patent or inventor's certificate, or Section 365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed.

<u>09 SEPTEMBER 1999</u>	Priority Claimed 優先権主張	
(Day/Month/Year Filed) (出願年月日)	<input checked="" type="checkbox"/> Yes はい	<input type="checkbox"/> No いいえ
<u> </u>	<input type="checkbox"/> Yes はい	<input type="checkbox"/> No いいえ
(Day/Month/Year Filed) (出願年月日)		

I hereby claim the benefit under Title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below.

<u> </u>	<u> </u>
(Application No.) (出願番号)	(Filing Date) (出願日)

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s), or Section 365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT International filing date of application.

<u> </u>
(Status: Patented, Pending, Abandoned) (現況: 特許許可済、係属中、放棄済)

<u> </u>
(Status: Patented, Pending, Abandoned) (現況: 特許許可済、係属中、放棄済)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Japanese Language Declaration

(日本語宣言書)

委任状：私は下記の発明者として、本出願に関する一切の手続きを米特許商標局に対して遂行する弁理士または代理人として、下記の者を指名いたします。
(弁理士、または代理人の指名及び登録番号を明記のこと)

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith: (list name and registration number)

Norman F. Oblon, Reg. No. 24,618; Marvin J. Spivak, Reg. No. 24,913; C. Irvin McClelland, Reg. No. 21,124; Gregory J. Maier, Reg. No. 25,599; Arthur I. Neustadt, Reg. No. 24,854; Richard D. Kelly, Reg. No. 27,757; James D. Hamilton, Reg. No. 28,421; Eckhard H. Kuesters, Reg. No. 28,870; Robert T. Pous, Reg. No. 29,099; Charles L. Gholz, Reg. No. 26,395; William E. Beaumont, Reg. No. 30,996; Jean-Paul Lavalleye, Reg. No. 31,451; Stephen G. Baxter, Reg. No. 32,884; Richard L. Treanor, Reg. No. 36,379; Steven P. Weihrouch, Reg. No. 32,829; John T. Goolkasian, Reg. No. 26,142; Richard L. Chinn, Reg. No. 34,305; Steven E. Lipman, Reg. No. 30,011; Carl E. Schlier, Reg. No. 34,426; James J. Kulbaski, Reg. No. 34,648; Richard A. Neifeld, Reg. No. 35,299; J. Derek Mason, Reg. No. 35,270; Surinder Sachar, Reg. No. 34,423; Jeffrey B. McIntyre, Reg. No. 36,867; William T. Enos, Reg. No. 33,128; Michael E. McCabe, Jr., Reg. No. 37,182; Bradley D. Lytle, Reg. No. 40,073, and Michael R. Casey, Reg. No. 40,294, with full powers of substitution and revocation.

書類送付先

Send Correspondence to:

OBLON, SPIVAK, MCCLELLAND, MAIER & NEUSTADT, P.C.
FOURTH FLOOR
1755 JEFFERSON DAVIS HIGHWAY
ARLINGTON, VIRGINIA 22202 U.S.A.

直接電話連絡先：(名前及び電話番号)

Direct Telephone Calls to: (name and telephone number)

(703) 413-3000

単独発明者または第一の共同発明者の氏名	1-00	Full name of sole or first joint inventor	Takashi MATSUMOTO
発明者の署名	日付	Inventor's signature	松本 尚
住所		Residence	1-170-2-203, Yayoi-cho, Inage-ku, Chiba-shi, Chiba 263-0022 Japan
国籍		Citizenship	Japan JPX
郵便の宛先		Post Office Address	same as above
第二の共同発明者の氏名		Full name of second joint inventor, if any	
第二の共同発明者の署名	日付	Second joint Inventor's signature	Date
住所		Residence	
国籍		Citizenship	
郵便の宛先		Post Office Address	

(第三以降の共同発明者についても同様に記載し、署名すること)

(Supply similar information and signature for third and subsequent joint inventors.)